WORKING WITH MULTIVARIATE POLYNOMIALS IN MAPLE

JEFFREY B. FARR AND ROMAN PEARCE

Abstract. We comment on the implementation of various algorithms in multivariate polynomial theory. Specifically, we describe a modular computation of triangular sets and possible applications. Next we discuss an implementation of the F_4 algorithm for computing Gröbner bases. We also give examples of how to use Gröbner bases for vanishing ideals in polynomial and rational function interpolation.

1. Introduction

Since Maple9, the ability of Maple to handle larger and more varied problems dealing multivariate polynomials has increased significantly. In fact a new package, `PolynomialIdeals`, was introduced in Maple9.5 and is described in [10]. In this paper we present enhancements, extensions and applications of these improvements. We first describe the computation of triangular sets, which in some applications provide a more efficient alternative to Gröbner bases, and we discuss a modular implementation. In the next section, we deal with an implementation of the so-called F_4 algorithm for computing Gröbner bases. This algorithm gives a substantial improvement over the Buchberger algorithm in practice. Finally, we describe a new Maple command to compute a Gröbner basis for the vanishing ideal of a set of multidimensional affine points and show how to use this command to solve multivariate polynomial and rational function interpolation problems.

2. Triangular Sets

It is well known that lexicographic Gröbner bases can have exceptionally large coefficients and that alternative triangular forms for polynomial systems offer substantial savings. In particular, Dahan and Schost [3] describe a representation in which the coefficient length is linear in the number of solutions of the system. This compares quite favorably to lexicographic Gröbner bases, for which the coefficient length is quadratic. Starting from a lexicographic basis, the Dahan-Schost form can be computed as follows.

Dahan-Schost Transform

Input $[g_1, \ldots, g_t]$ a sorted (ascending) lexicographic Gröbner basis for a zero dimensional ideal with $x_1 < \cdots < x_n$.
Output $[h_1, \ldots, h_n]$ the Dahan-Schost form

Date: 18 May, 2005.
Research supported by NSERC and MITACS NCE of Canada.

$h_1 \leftarrow g_1$
$d_1 \leftarrow g_1' \,/\, \gcd(g_1, g_1')$

for i **from** 2 **to** n **do**
  select the smallest g_j with leading monomial x_i^k
  $h_i \leftarrow (d_1 \cdot g_j) \mod g_1$
end loop

return $[h_1, \ldots, h_n]$

Our goal is to compute this representation without first computing a lexicographic Gröbner basis. We have developed a probabilistic modular method based on the FGLM algorithm for converting Gröbner bases [7].

The FGLM algorithm counts up through the monomials of the polynomial ring while testing their normal forms for linear dependence. Each dependency produces a linear combination of monomials which is in the ideal, or equivalently, is an element of the desired Gröbner basis. At the end of the algorithm one can express this basis as the solution of a linear system $AX = B$, where the columns of A consist of the independent normal forms and the columns of B are the dependent ones.

The first step of our algorithm constructs this system for a lexicographic Gröbner basis; however, all of the linear dependency checking is done modulo a small batch of primes. The system is then solved modulo batches of primes and Chinese remaindering is applied, resulting in the image of a lex Gröbner basis modulo the product of the primes. The Dahan-Schost transformation converts this image into that of a triangular set, and rational reconstruction recovers the result. The advantage of constructing the linear system exactly is that should rational reconstruction fail, additional primes can be added with very little recomputation.

The pseudocode below also contains two other optimizations which make the algorithm more efficient. First, we do not compute every element of the lexicographic Gröbner basis; we solve only for the elements which are needed to construct the triangular set. Secondly, the algorithm reconstructs polynomials one at a time, so that in practice some computations are done using a smaller modulus.

We have implemented this algorithm in Maple using the hardware datatypes and compiled routines in the `LinearAlgebra:-Modular` package. The float[8] datatype, which uses 25-bit primes, tends to give the best performance overall. The initial linear dependency calculations use a batch of ten primes, which results in a probability of error which is typically less than 10^{-50}.

Table 2 compares our ability to compute triangular sets versus lexicographic Gröbner bases, starting from a total degree Gröbner basis. This is significant because there are algorithms for primality testing, primary decomposition, and radical computation which currently use lexicographic Gröbner bases but could be adapted to use triangular sets instead [8].

Multimodular Triangular Set

Input G a Gröbner basis for a zero-dimensional ideal I
$[x_1, \ldots, x_n]$ a list of variables
Output $T = [t_1, \ldots, t_n]$ the Dahan-Schost form

\# construct $AX = B$
$M_G \leftarrow$ a vector of the monomials not reducible by G
$M_A \leftarrow$ an $|M_G| \times 1$ vector
$M_B \leftarrow$ an $n \times 1$ vector
$A \leftarrow$ an $|M_G| \times |M_G|$ matrix
$B \leftarrow$ an $|M_G| \times n$ matrix
$border \leftarrow \{\}$
$m \leftarrow 1$
while $m \neq FAIL$ **do**
  $r \leftarrow NormalForm(m, G)$
  $C \leftarrow$ the coefficients of r, with $C \cdot M_G = r$
  if C is independent of the current columns of A
    write C into the next column of A
    write m into the next column of M_A
  else if C is dependent and $m = x_i^k$ then
    write C into the next column of B
    write m into the next column of M_B
    $border \leftarrow border \cup \{m\}$
  $m \leftarrow$ the next monomial not divisible by a *border* element
end loop

\# solve $AX = B$
$X \leftarrow$ an $|M| \times 1$ zero vector
$N \leftarrow 1$
$i \leftarrow 1$
while t_n not constructed **do**
  choose a batch of new primes $\{p_1, \ldots, p_s\}$
  $\{X_1, \ldots, X_s\} \leftarrow$ the solution of $AX = B \mod p_j$
  $X \leftarrow ChineseRemainder([X, N], [X_1, p_1], \ldots, [X_s, p_s])$
  $N \leftarrow (\prod_{j=1}^{s} p_j)\, N$
  while $i \leq n$ and no failure has occurred **do**
    if $i = 1$ **then**
      $t_1 \leftarrow RationalReconstruction(M_B[i] - M_A \cdot X_i, N)$
      if $t_1 \neq FAIL$ **then**
        $d_1 \leftarrow t_1' \,/\, \gcd(t_1, t_1')$
        $i \leftarrow i + 1$
    else
      $t_i \leftarrow RationalReconstruction(d_1\,(M_B[i] - M_A \cdot X_i) \mod t_1, N)$
      if $t_i \neq FAIL$ **then**
        $i \leftarrow i + 1$
  end loop
end loop
return $T = [t_1, \ldots, t_n]$

System	*Dim*	T_{digits}	T_{sec}	L_{digits}	L_{sec}
Katsura-5	32	35	1.03	576	1.02
Katsura-6	64	76	4.25	2016	4.93
Katsura-7	128	179	32.3	10892	248.33
Katsura-8	256	379	364	big	9708
Katsura-9	512	859	5220	–	–

The first column, *Dim*, is the size of the matrix A or, equivalently, the dimension of the quotient ring as a vector space. The columns T_{digits} and L_{digits} are the sizes of the coefficients in the triangular set and the lexicographic Gröbner basis, respectively. The triangular set computations were done using 64-bit Maple10 on an Opteron 248 2.2 GHz with 4 GB of RAM. The lexicographic computations used the computer algebra system Magma 2.11-12 on an Opteron 250 2.4 GHz with 8 GB RAM. Magma uses sparse p-adic lifting and floating point arithmetic with moduli up to 24 bits [11], so the linear algebra implementations are comparable.

We are presently working to integrate this algorithm into Maple. Our goal is to modify all possible algorithms in the `PolynomialIdeals` package so that they use triangular sets instead of lexicographic Gröbner bases.

3. The F_4 Algorithm

Computing a Gröbner basis is often a first step towards solving or working with a system of polynomial equations. It can also be the most difficult step since the polynomials lack any particular mathematical structure. The F_4 algorithm for computing Gröbner bases was first described in [6], and the current implementation in Magma is among the fastest widely available routines for computing Gröbner bases [11].

One way to visualize the F_4 algoritm is to consider the reduction of a single S-polynomial in the Buchberger algorithm. For example, let $G = [x^2 + y, xy^2 - xy, y^3 - 1]$ and consider the syzygy $S_{1,2} = x^2y + y^3$ under graded lex order. In the division algorithm, we will reduce $S_{1,2}$ first by subtracting $y\,G_1$ and then by subtracting G_3, as shown below.

$$\begin{array}{lcll} x^2y + y^3 & \rightarrow & x^2y + y^3 - y\,(x^2+y) & = y^3 - y^2 \\ & \rightarrow & y^3 - y^2 - (y^3 - 1) & = -y^2 + 1 \end{array}$$

The key observation is that this reduction process is equivalent to a matrix triangularization. In the example below, the columns of the matrix correspond to the monomials $[x^2y, y^3, y^2, 1]$, while the rows contain $S_{1,2}$, $y\,G_1$, and G_3, respectively. Examining the reduced matrix on the right, we find one new pivot belonging to $y^2 - 1$.

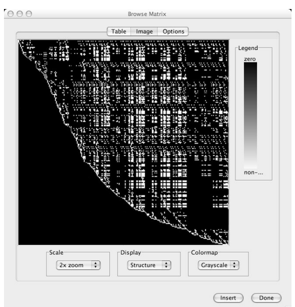
$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

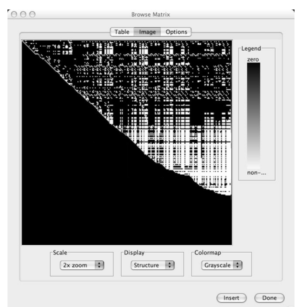
From this perspective we can see what is "wrong" with the Buchberger algorithm. It selects syzygies one by one, and for each one it triangularizes an entire matrix.

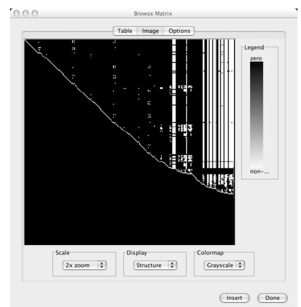
In general these matrices are big, and it is not hard to imagine that they may have many rows in common.

The F_4 algorithm consists of a very simple improvement: one runs the Buchberger algorithm but at each step selects multiple syzygies. They are placed into a common matrix along with any rows that are needed for the reduction process, and this matrix is triangularized. The rows with new pivots correspond to new polynomials, which are then added to the basis.

Faugère discusses various strategies for the F_4 algorithm in [6]. In particular, one should select all of the syzygies of smallest degree at each step of the algorithm, and reuse rows from previously reduced matrices where possible. To this we contribute the following observation. Below is a matrix from a step in the computation for cyclic-6. On the left is the original system, followed by its row echelon form and reduced row echelon form, respectively.







All of these matrices are sparse, however the reduced row echelon form is extremely sparse. We suggest that if one is to reuse rows from previous matrices frequently, it is worth the extra cost to reduce each matrix to reduced row echelon form. Computer experiments have borne out this hypothesis. In homogeneous computations, in which the degrees of the syzygies increase monotonically, this strategy produces smaller matrices over the course of the algorithm, typically on the order of 15 to 20 percent.

This potential improvement is not fully realized, however, because a second improvement, computing modulo a number of primes, offsets some of the advantage. In such an algorithm, the matrix will be reduced modulo a number of primes until the desired rows can be recovered using Chinese remaindering and rational reconstruction. Over algebraic function fields sparse rational function interpolation will also be used so that the cost of recovering each row becomes significantly higher.

In any case, the best strategy seems to be a hybrid approach. That is, after the initial reductions modulo a prime, one can identify rows with new pivots and further reduce them using the rest of the matrix. These sparse rows are easier to reconstruct, and as a side effect one computes the reduced Gröbner basis automatically.

Since conducting these experiments, we have been working on a more robust implementation of F_4 for Maple. Early prototypes have shown that significant improvements are possible relative to the implementation of Buchberger's algorithm in Maple10. Our initial focus is on rational coefficients and the integers modulo a prime, although algebraic function fields will be supported in the final version.

The most pressing need at this time is enhanced algorithms and data structures for sparse linear algebra in Maple.

4. Vanishing ideals and multivariate interpolation

In some situations we may be given a collection of points $V \subseteq \mathbb{F}^n$, where $\mathbb{F}$ is any field, and be asked to find a (reduced) Gröbner basis with respect to a given term order for the vanishing ideal defined by

$$\mathbf{I}(V) = \{f \in \mathbb{F}[x_1, \ldots, x_n] : f(P) = 0 \text{ for all } P \in V\}.$$

Unlike many Gröbner basis problems, this one is not hard to solve in the sense that there exist algorithms that produce a solution in polynomial (in the number of points and in the number of variables) time. Such an algorithm, based on Gauss elimination, was first given by Buchberger and Möller [2]. Subsequent improvements and generalizations include [9, 1]. These algorithms may be viewed as multivariate analogues of univariate Lagrange interpolation. Recently, an alternate solution, analogous to univariate Newton interpolation, was presented [5] and is included in the `PolynomialIdeals` package in Maple10 with the command `VanishingIdeal`. We illustrate the command with the following example. The output is of type `'PolynomialIdeal'`, and the generators that are displayed are the Gröbner basis elements.

Example 1.

```
> with(PolynomialIdeals):
> V:=[[1,-1],[1,1],[1,3],[2,-1],[2,1],[2,3],[4,-1],[4,1],[4,3],[0,0]]:
> VanishingIdeal( V, [x,y], tdeg(y,x) );
```

$$\langle x^4 - 7x^3 + 14x^2 - 8x, x^3y - 7x^2y + 14xy - 8y, 8y^3 + 3x^3 - 24y^2 - 21x^2 - 8y + 42x\rangle$$

While the computation of Gröbner bases for vanishing ideals is an interesting study in its own right, it is also implicitly present in several applications. We present here several algorithms that we have implemented and are working to integrate into Maple. First, suppose that for each point $P_i \in V$ we have a corresponding value $r_i \in \mathbb{F}$. Then the multivariate interpolation problem is to find the "smallest" polynomial f such that $f(P_i) = r_i$.

This problem is not trivial. In particular, there is no single set of m monomials that can serve as a basis for an interpolation space for m points in $\mathbb{F}^n$. For instance, take the set V of 10 points in $\mathbb{Q}^2$ in Example 1. Suppose the interpolant desired is the one with smallest total degree (although any other monomial order is also acceptable). It is incorrect to assume that the set of 10 smallest monomials in $\mathbb{Q}[x, y]$, namely $\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3\}$, may serve as a basis for an interpolation space in the case in which the points in V are the independent points. In fact these monomials cannot form a basis because they are linearly dependent on the points of V; *i.e.*, $g = 8y^3 + 3x^3 - 24y^2 - 21x^2 - 8y + 42x = 0$ for every $P \in V$ as the presence of g in the Gröbner basis for $\mathbf{I}(V)$ indicates.

Hence, an appropriate interpolation space must be found for each set of independent points before the actual interpolation takes place. The monomial basis of an ideal with respect to a certain term order is the set of all monomials not divisible by the leading term of any polynomial in the Gröbner basis of the ideal with respect to that term order, and the members of the monomial basis are linearly independent on the points of V. So by computing the Gröbner basis above, we have actually already found the desired interpolation space: $\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, x^2y^2\}$.

The above explanation means that we can find the interpolating polynomial f by finding the Gröbner basis of the vanishing ideal of the set of points (P_i, r_i). The monomial order that must be used is an elimination order for the new variable corresponding to the r_i; the original term order is used on the remaining variables. We have implemented this algorithm as `MultivariateInterpolation`.

```
> r := [6,10,-10,5,13,-11,9,25,41,9]:
> MultivariateInterpolation( V, r, [x,y], tdeg(y,x) );
```

$$x^2y^2 - 4xy^2 + 2xy + 2x + 9$$

Both `VanishingIdeal` and `MultivariateInterpolation` have versions that allow the user to work modulo a prime.

```
> VanishingIdeal(V,[x,y],tdeg(y,x)) mod 7;
> MultivariateInterpolation( V, r, [x,y], tdeg(y,x) ) mod 7;
```

$$\langle x^4 + 6x, 3x^3 + y^3 + 6y + 4y^2, x^3y + 6y\rangle$$

$$x^2y^2 + 3xy^2 + 2xy + 2x + 2$$

Another natural interpolation problem involving multivariate polynomials is rational function interpolation. While there are several ways to approach this problem, a recent result [4] provides the most complete solution using a Gröbner basis approach. The first part of the solution is to find the interpolation space, which we accomplish as before. But the main difficulty that this algorithm overcomes is in determining a suitable term order for the vanishing ideal computation. The solution given, though, requires using modules of rank two over the polynomial ring rather than using polynomial ideals. However, due to the flexibility of MAPLE's `'matrix'` term order, we can work (carefully!) within the `PolynomialIdeals` package and not call on the more expensive machinery for modules. Once again, computing modulo a prime is allowed.

Example 2. We consider the set V below of eight points from $\mathbb{Q}^2$. The monomial basis of $\mathbf{I}(V)$ with respect to $\mathrm{tdeg}(x,y)$ is $\mathcal{B} = \{1, y, x, y^2, xy, x^2, y^3, xy^2\}$. The `MultivariateRationalInterpolation` command requires one additional parameter, t_1. This parameter indicates the size of the numerator; specifically, the numerator must be in the linear span of the first t_1 elements of $\mathcal{B}$. The denominator must be in the linear span of the first $|V| - t_1 + 1$ elements of $\mathcal{B}$. Since one of the coefficients in the numerator or denominator may be fixed (we fix the denominator to be monic), there are $|V|$ coefficients to determine. In this example we take $t_1 = 5$, so the numerator is in the span of $\{1, y, x, y^2, xy\}$ and the denominator in $\{1, y, x, y^2\}$.

```
> V:=[[2,3],[1,0],[1,2],[2,1],[3,0],[2,2],[3,4],[0,2]]:
> r:=[10,10,4,-8,-18,22,-16,2,0]:
> MultivariateRationalInterpolation(V,r,5, [x,y], tdeg(x,y));
```

$$\frac{(xy - 2y^2 + 6x + 2y + 4)}{(y^2 - 3y + 1)}$$

If we keep V and r the same but change to a weighted term order, we obtain a different interpolant.

```
> MultivariateRationalInterpolation(V,r,5, [x,y], wdeg([2,1],[x,y]));
```

$$\frac{-2(43y^3 - 528x - 200y^2 + 404y - 352)}{(151y^2 - 456y + 176)}$$

To this point we have implicitly assumed that the points in V are distinct. Of course this is not always the case. Multiplicity in a multivariate setting has various meanings, but even under a fairly broad algebraic definition each of these algorithms can be modified to handle nontrivial multiplicities. For the rational function interpolation problem, the extreme case of having one point with multiplicity is, in fact, multivariate Padé approximation. We illustrate with a final example.

Example 3. Without loss of generality we may assume that the point in question is the origin. The set `MB` denotes the monomial basis for a monomial ideal I and defines multiplicity in the sense described in the following paragraph. As in `MultivariateRationalInterpolation`, the parameter "10" gives the size of the numerator.

```
> h := mtaylor( sin(x+y) + cos(x+y), [x,y], 8);
> MB:=[1,x,y,x^2,xy,y^2,x^3,x^2y,xy^2,y^3,x^4,x^3y,x^2y^2,xy^3,y^4]:
> hpade := MultivariatePade(h, MB, 10, [x,y,NewVar],
>              'matrix'([[1,1,1],[0,1,1],[0,0,1]], [x,y,NewVar]), 0);
```

$$h := 1 + x + y - 1/2x^2 - xy - 1/2y^2 \cdots - 1/240y^5x^2 - 1/720y^6x - 1/5040y^7$$

$$hpade := \frac{(-4xy + 5x^3 - 6x - 4x^2 + 10x^2y + 5xy^2)}{2(xy + x^2 - 3x)}$$

The measure of "closeness" that the approximant must satisfy is the so-called weak interpolation criterion; that is, if a/b is the approximant, then $b \cdot h - a \in I$. In other words the coefficient for each element in `MB` in the polynomial $bh - a$ must be zero. We verify that this is so by using the trailing coefficient command, `tcoeff`.

```
> g := simplify( denom(hpade)*h - numer(hpade) ):
> tcoeff(g,[y,x],'tm'):
> tm;
```

$$x^5$$

References

[1] J. Abbott, A. Bigatti, M. Kreuzer and L. Robbiano, Computing ideals of points, *J. Symbolic Comput.* **30** (2000), 341-356.

[2] B. Buchberger and H.M. Möller, The construction of multivariate polynomials with preassigned zeros, *Computer algebra, EUROCAM '82*, pp. 24-31, Lecture Notes in Comput. Sci., vol. 144, Springer, Berlin-New York, 1982.

[3] Xavier Dahan and Eric Schost, Sharp estimates for triangular sets, Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (Santander, 2004), 103–110.

[4] Jeffrey B. Farr and Shuhong Gao, Gröbner bases and generalized Padé approximation, to appear in *Math. Comp.*

[5] Jeffrey B. Farr and Shuhong Gao, Computing Gröbner bases for vanishing ideals of finite sets of points, *preprint.*

[6] Jean-Charles Faugère, A new efficient algorithm for computing Gröbner Bases (F_4), Effective methods in algebraic geometry (Saint-Malo, 1998), *J. of Pure Appl. Algebra*, **139**, (1999), no. 1-3, 61–88.

[7] J.C. Faugère, P. Gianni, D. Lazard and T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, *J. Symbolic Comput.* **16** (1993), 329–344.

[8] P. Gianni, B. Trager and G. Zacharias, Gröbner bases and primary decompositions of polynomial ideals, *J. Symbolic Comput.* **6** (1988), pp. 149–167.

[9] M. G. Marinari, H.M. Möller and T. Mora, On multiplicities in polynomial system solving, *Trans. Amer. Math. Soc.* **348** (1996), no. 8, 3283-3321.

[10] M. B. Monagan and R. Pearce, The PolynomialIdeals Maple package, Proceedings of the 2004 Maple Summer Workshop.

[11] A. Steel, Gröbner Basis Timings Page http://magma.maths.usyd.edu.au/users/allan/gb

Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada V5A 1S6 *E-mail address*: {jfarr, rpearcea}@cecm.sfu.ca